REMARKS/ARGUMENTS

Claims 1-22 remain in this application for further consideration. In light of the explanation set forth below, regarding the teaching of the cited art, applicants believe that the claims are allowable. Applicants request further consideration and allowance of the forgoing claims.

I. Rejection of Claims 1-22 under 35 U.S.C. 102(b).

Claims 1-22 were rejected under 35 U.S.C. 102(b) as being anticipated by *Larus*, Whole Program Paths (May 1999) (hereinafter "Larus"). In applicants' prior amendment, applicants pointed out that Larus does not teach, along with other elements, the "removal of less frequently occurring data access sequences from a trace file." In response, the current Office Action iterates that this limitation is taught by the recitation "[a]t this point, non-terminals B and C are only used once and SEQUITUR eliminates them." (*Larus*, at 261). As more fully set forth below, the SEQUITUR algorithm (page 261) of Larus does not teach the elements propounded in the Office Action.

A. Elements of the Independent Claims That Are Not Taught By Larus

Applicants' independent claim 1 specifically recites the following elements that are not taught or suggested by the Larus reference:

> "using the identified sequences to create a ***modified trace file*** by ***removing less frequently occurring data access sequences*** from the trace file." (Emphasis added).

Applicants' independent claim 10 specifically recites the following elements that are not taught or suggested by the Larus reference:

> "when the frequently occurring data access pattern follows another frequently occurring data access pattern, ***updating a data structure to reflect that the data access pattern follows the other data access pattern***." (Emphasis added).

Applicants' independent claim 17 specifically recites the following elements that are not taught or suggested by the Larus reference:

> "a stream flow graph structured to store data that indicates ***a frequency that a data access sequence follows another data access sequence***." (Emphasis added).
>
> "a ***pre-fetcher*** configured to use the data access information and the stream flow graph to ***fetch data elements into memory*** for use by the executing computer program." (Emphasis added).

Applicants' independent claim 21 specifically recites the following elements that are not taught or suggested by the Larus reference:

> "a ***cache memory manager*** coupled to the stream flow graph database and the data access sequence database, wherein the ***cache memory manager is configured to arrange data elements of a repetitively accessed data stream*** in a cache using information from the two databases." (Emphasis added).

B. <u>*Larus*, Whole Program Paths (May 1999)</u>.

Applicants assert that Larus does not teach or otherwise suggest at least the above-recited elements of the independent claims. Below is a detailed explanation of the portions of the Larus reference cited in the Office Action. Applicants believe that this explanation will shed light on the distinction between the Larus reference and the claims of the present invention.

1. <u>Section 3.1 - SEQUITUR Algorithm</u>

Section 3.1 of Larus, does not teach "using the identified sequences to create a modified trace file by removing less frequently occurring data access sequences from the trace file." SEQUITUR is a formula used to represent a data access sequence. SEQUITUR uses formulas provide representations of the sequence or string when elements are repeated. However, the entire string or sequence is still associated with the formula. Stated another way, the entire data access sequence is "unwound" when the formulas are calculated. Elements of the data access sequence are not eliminated.

The SEQUITUR algorithm follows two rules. The first rule is referred to as the Digram Uniqueness Property Rule. A digram is a pair of consecutive symbols. For example, in the sequence (abca), the symbols (ab) may be considered a digram. If two of the same digrams exist, SEQUITUR replaces both occurrences of the digram with a non-terminal symbol. For example, digram uniqueness property rule may work as follows:

S = abca	SEQUITUR does not apply because there is not two or more digrams

S = abcab
S = AcA
A = ab
	SEQUITUR applies because two (ab)s exist

In the above example, the symbols (ab) occurs twice. SEQUITUR replaces the symbols (ab) with the symbol (A). The symbol (A) is then set equal to (ab). In this manner, SEQUITUR can reduce the size of a string through a formula representation when repeating elements exist. However, when the formula is calculated, the entirety of the string still exists. SEQUITUR does not remove any data access sequences; it merely represents the data access sequences in a formula. For example, in the above, symbols (ab) are not removed. They still exist in the formula (A=ab). The sequence is merely represented in a different manner through SEQUITUR.

The second rule of SEQUITUR is referred to as the Utility Property Rule. The Utility Property Rule states that all non-terminal symbols (capitalized letters) in a grammar must be referenced more than once by other rules (i.e. A=ab, B=Ac, C=Ad, or D=BC). SEQUITUR exchanges a rule referenced only once with the rules right side. For example, before the Utility Property Rule is applied, a formula may be as follows:

S =-abcabdabcabd
S = DD
A = ab
B = Ac
C = Ad
D = BC

When the rule D=BC is expanded, the string ultimately becomes S=abcabdabcabd. However, when SEQUITUR applies the Utility Property Rule to the above algorithm, the algorithm is as follows:

S = abcabdabcabd
S = DD
A = ab
D = AcAd

The formula represents the same string. When D=AcAd is expanded, the string ultimately becomes S=abcabdabcabd. As is shown in the above example, non-terminal symbols (B) and (C) are no longer used in the formula. Their corresponding symbols (Ac) and (Ad) are then moved down into symbol (D). In accordance with the utility property rule, non-terminals (B) and (C) were only used once in the formula; therefore, SEQUITUR exchanges these symbols by including their equation in symbol (D). Such a procedure makes the algorithm more efficient for expanding the sequence. However, the above does not indicate that elements of the ***data access sequence*** are eliminated. Both the examples above indicate that S = abcabdabcabd. Elements of the sequence are not removed. The utility property rule merely indicates that the equation for "unwinding" the sequence is written in a certain manner. The data access sequence, however, is the same regardless of the formula used to represent it. Accordingly, Larus does not teach "using the identified sequences to create a modified trace file by removing less frequently occurring data access sequences from the trace file," as specifically recited in claim 1.

2. FIGURES 1, 2 and 7.

FIGURES 1-3 of Larus, do not teach that "when the frequently occurring data access pattern follows another frequently occurring data access pattern, updating a data structure to reflect that the data access pattern allows the other data access patterns." FIGURES 1 and 2 of Larus are functional block digrams. FIGURE 1 depicts that the Whole Program Path is sent for analysis to find performance bottlenecks or program errors.

FIGURE 2 depicts a SEQUITUR grammar and a whole program path that represents the SEQUITUR grammar. The Whole Program Path represents the SEQUITUR grammar through arrows. For example, referring to FIGURE 2, the SEQUITUR grammar C = BB is represented in the Whole Program Path by a (C) node having two arrows pointing to the (B) node. In this manner, the entire SEQUITUR sequence may be represented through Whole Program Paths.

FIGURE 7 represents an algorithm or program for finding minimal hot subpaths whose length is between MinStringLength and MaxStringLength and cost greater than MinCost. FIGURE 7, merely teaches finding sequences that have a certain length and occur a certain number of times. Applicants can find no teaching or suggestion in the aforementioned figures (and accompanying text) of a teaching that "when the frequently occurring data access pattern follows another frequently occurring data access pattern, updating a data structure to reflect that the data access pattern follows the other data access patterns." Applicants respectfully request that the Examiner point out with specificity the specific text or digram of Larus that teaches this element. If this element is not specifically taught or suggested, applicants request a notice of allowance of claim 10.

3. Section 3.2 - SEQUITUR Enhancement.

Section 3.2 of Larus does not teach "a stream flow graph structured to store data that indicates a frequency that a data access sequence follows another data access sequence." Also, Section 3.2 of Larus does not teach "a pre-fetcher configured to use the data access information and the stream flow graph to fetch data elements into memory for use by the executing computer program." Section 3.2 of Larus pertains to enhancing the SEQUITUR algorithm by looking ahead a single symbol before introducing a new digram rule. When the algorithm does not look ahead one symbol, representation of the same occurrences can vary, because rules introduced while processing a first occurrence may change the sequences of reductions applied to a second occurrence.

Applicant can find no teaching or suggestion of the elements of independent claim 17. Applicant respectfully requests that the Examiner point out with specificity in Section 3.2 the specific text or digram of Larus that teaches this element. If this element is not specifically taught or suggested, applicants request a notice of allowance of claim 17.

4. FIGURE 2 and Whole Program Paths

Neither FIGURE 2 of Larus nor the discussion pertaining to Whole Program Paths teach or otherwise suggest "a cache memory manager coupled to the stream flow graph database and the data access sequence database, wherein the cache memory manager is configured to arrange data elements of a repetitively accessed data stream in a cache using information from the two databases." Moreover, such elements are not inherent in the Larus reference.

As previously stated, FIGURE 2 depicts a SEQUITUR grammar and a whole program path the represents the SEQUITUR grammar. The Whole Program Path represents the SEQUITUR grammar through arrows. For example, the SEQUITUR grammar C = BB is represented in the Whole Program Path by a (C) node having two arrows pointing to the (B) node. In this manner, the entire SEQUITUR sequence may be represented. There is no teaching in FIGURE 2 of a cache memory, let alone, any teaching of a cache memory manager configured to arrange data elements of a repetitively accessed data stream in a cache using information from the stream flow graph database and the data access sequence database.

Larus does not inherently teach the elements of claim 21. "To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill." *In re Robertson*, 49 USPQ2d 1949, 1950-51 (Fed. Cir. 1999). Applicants can find no mention of a ***cache memory manager*** in the entire Larus document. In addition, applicants cannot find any teaching or suggestion of ***a cache memory manager that is configured to arrange data elements of a repetitively accessed data stream in a cache using information from a stream flow graph database and a data access sequence database***. Applicants respectfully requests that the Examiner specifically point out some teaching or suggestion in Larus with regard to independent claim 21 or clarify the aforementioned inherency rejection. In the absence of one of the two, applicants respectfully request notice of allowance of claim 21.

C. Claims 2-9, 11-16, 18-20 and 22

Claims 2-9, 11-16, 18-20 and 22 ultimately depend from claims 1, 10, 17, and 21, respectively. Applicants assert that each of the claims recite allowable subject matter as more fully set forth in applicants August 27, 2004 Response. Moreover, claims 1, 10, 17, and 21 are clearly allowable for the reasons set forth above, and therefore, applicants assert the claims 2-9, 11-16, 18-20 and 22 are allowable for at least those same reasons.

In view of the foregoing, all pending claims are believed to be allowable and the application is in condition for allowance. Therefore, a Notice of Allowance is respectfully requested. Should the Examiner have any further issues regarding this application, the Examiner is requested to contact the undersigned attorney for the applicant at the telephone number provided below.

Respectfully submitted,

MERCHANT & GOULD P.C.

Ryan T. Grace
Registration No. 52,956
Direct Dial: 206.342.6258

MERCHANT & GOULD P.C.
P. O. Box 2903
Minneapolis, Minnesota 55402-0903
206.342.6200

27488
PATENT TRADEMARK OFFICE